Ensemble learning

**Basic idea:** if one classifier works well, why not use multiple classifiers!
Ensemble learning

**Basic idea:** if one classifier works well, why not use multiple classifiers!

```
example to label
```

```
model 1  
model 2  
...  
model m  
```

### Testing

```
model 1  
prediction 1  
model 2  
prediction 2  
...  
prediction m  
```

Idea 4: boosting

```
<table>
<thead>
<tr>
<th>Data</th>
<th>Label</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
</tbody>
</table>

```

```
<table>
<thead>
<tr>
<th>Data</th>
<th>Label</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0.1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.4</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0.1</td>
</tr>
</tbody>
</table>

```

```
<table>
<thead>
<tr>
<th>Data</th>
<th>Label</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.05</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0.5</td>
</tr>
</tbody>
</table>
```

“Strong” learner

**Given**
- a reasonable amount of training data
- a target error rate $\varepsilon$
- a failure probability $p$

**A strong learning algorithm** will produce a classifier with error rate $< \varepsilon$ with probability $1-p$

“Weak” learner

**Given**
- a reasonable amount of training data
- a failure probability $p$

**A weak learning algorithm** will produce a classifier with error rate $< 0.5$ with probability $1-p$

Weak learners are much easier to create!
Need a weak learning algorithm that can handle weighted examples.
Boosting

Weights:

Examples: E1 E2 E3 E4 E5

- decrease the weight for those we're getting correct
- increase the weight for those we're getting incorrect

Learn another weak classifier:

Weights:

Examples: E1 E2 E3 E4 E5

- decrease the weight for those we're getting correct
- increase the weight for those we're getting incorrect
Classifying

- weak 1
  - prediction 1
  - weighted vote based on how well they classify the training data
  - weak 2
    - prediction 2
    - weak_2_vote > weak_1_vote since it got more right

Notation

- \( x_i \): example i in the training data
- \( w_i \): weight for example i, we will enforce:
  - \( w_i \geq 0 \)
  - \( \sum_i w_i = 1 \)
- classifier_k(x): +1/-1 prediction of classifier k example i

AdaBoost: train

- for k = 1 to iterations:
  - classifier_k = learn a weak classifier based on weights
  - calculate weighted error for this classifier
    - \( \epsilon_k = \sum_i w_i \cdot \text{if label, classifier_k(x_i)} \)
  - calculate “score” for this classifier:
    - \( \alpha_k = \frac{1}{2} \cdot \log \left( \frac{1 - \epsilon_k}{\epsilon_k} \right) \)
  - change the example weights
    - \( w_i = \frac{1}{Z} \cdot w_i \cdot \exp(-\alpha_k \cdot \text{if label, classifier_k(x_i)} \))

What does this say?
AdaBoost: train

classifier_k = learn a weak classifier based on weights

weighted error for this classifier is:

$$e_k = \sum_{i=1}^{n} w_i \cdot \mathbb{1}[\text{label}_i \neq \text{classifier}_k(x_i)]$$

-prediction
did we get the example wrong
-weighted sum of the errors/mistakes

What is the range of possible values?

Between 0, if we get all examples right, and 1, if we get them all wrong
-weighted sum of the errors/mistakes

“score” or weight for this classifier is:

$$\alpha_k = \frac{1}{2} \log \left( \frac{1-e_k}{e_k} \right)$$

What does this look like (specifically for errors between 0 and 1)?

- ranges from 1 to -1
- errors of 50% = 0
AdaBoost: classify

classify(x) = \text{sign} \left( \sum_{k=1}^{a} \alpha_k \cdot \text{classifier}_k(x) \right)

What does this do?

The weighted vote of the learned classifiers weighted by \( \alpha \) (remember \( \alpha \) varies from 1 to -1 training error)

What happens if a classifier has error >50%

AdaBoost: classify

classify(x) = \text{sign} \left( \sum_{k=1}^{a} \alpha_k \cdot \text{classifier}_k(x) \right)

AdaBoost: train, updating the weights

update the example weights

\[ w_i = \frac{1}{Z} w_i \exp(-\alpha_k \cdot \text{label}_i \cdot \text{classifier}_k(x_i)) \]

Remember, we want to enforce:

\[ w_i \geq 0 \]
\[ \sum_{i=1}^{a} w_i = 1 \]

\( Z \) is called the normalizing constant. It is used to make sure that the weights sum to 1

What should it be?
AdaBoost: train

update the example weights

\[ w_i = \frac{1}{Z} w_i \exp(-\alpha_k \cdot \text{label} \cdot \text{classifier}_k(x_i)) \]

Remember, we want to enforce:

\[ w_i \geq 0 \]
\[ \sum_i w_i = 1 \]

normalizing constant (i.e. the sum of the “new” \( w_i \):

\[ Z = \sum_i w_i \exp(-\alpha_k \cdot \text{label} \cdot \text{classifier}_k(x_i)) \]

What does this do?

Note: only change weights based on current classifier (not all previous classifiers)
AdaBoost: train

update the example weights

\[ w_i = \frac{1}{Z} w_i \exp(-\alpha_k \cdot \text{label} \cdot \text{classifier}(x_i)) \]

What does the \( \alpha \) do?

If the classifier was good (<50% error) \( \alpha \) is positive:
- trust classifier output and move as normal

If the classifier was back (>50% error) \( \alpha \) is negative:
- classifier is so bad, consider opposite prediction of classifier

Does this look like anything we've seen before?
**AdaBoost justification**

update the example weights

\[ w_i = \frac{1}{Z} w_i \exp(-\alpha \cdot \text{label} \cdot \text{classifier}(x_i)) \]

Exponential loss!

\[ l(y, y') = \exp(-yy') \]

AdaBoost turns out to be another approach for minimizing the exponential loss!

---

**Other boosting variants**

![Diagram of boosting variants]

- Adaboost: \( e^{-3(y \cdot x)} \)
- Logitboost
- Brownboost
- 0-1 loss

---

**Boosting example**

Start with equal weighted data set

- weak learner = line

What would be the best line learned on this data set?

\( h \Rightarrow p(\text{error}) = 0.5 \) it is at chance
This one seems to be the best.

This is a ‘weak classifier’; it performs slightly better than chance.

How should we reweight examples?

What would be the best line learned on this data set?
The strong (non-linear) classifier is built as the combination of all the weak (linear) classifiers.

**AdaBoost: train**

for \( k = 1 \) to iterations:
- classifier\(_k\) = learn a weak classifier based on weights
- weighted error for this classifier is:
- “score” or weight for this classifier is:
- change the example weights

What can we use as a classifier?

- Anything that can train on weighted examples
- For most applications, must be fast!
  Why?
AdaBoost: train

for k = 1 to iterations:
  - classifier_k = learn a weak classifier based on weights
  - weighted error for this classifier is:
  - “score” or weight for this classifier is:
  - change the example weights

- Anything that can train on weighted examples
- For most applications, must be fast!
- Each iteration we have to train a new classifier

Boosted decision stumps

One of the most common classifiers to use is a decision tree:
- can use a shallow (2-3 level tree)
- even more common is a 1-level tree
  - called a decision stump
  - asks a question about a single feature

What does the decision boundary look like for boosted decision stumps?

- Linear classifier!
  - Each stump defines the weight for that dimension
  - If you learn multiple stumps for that dimension then it’s the weighted average
Boosting in practice

Very successful on a wide range of problems

One of the keys is that boosting tends not to overfit, even for a large number of iterations

Using <10,000 training examples can fit >2,000,000 parameters!

Adaboost application example: face detection

Adaboost application example: face detection

Rapid Object Detection using a Boosted Cascade of Simple Features

Paul Viola
viola@eecs.berkeley.edu
Mitsubishi Electric Research Labs
201 Broadway, 8th FL
Cambridge, MA 02139

Michael Jones
mjones@eecs.berkeley.edu
Cerrpix CRL
One Cambridge Center
Cambridge, MA 02142

Rapid object detection using a boosted cascade of simple features

© 2001 6.867 Class Notes
Draft of 6.867 Class Notes, © 2001, Massachusetts Institute of Technology
To give you some context of importance:

The anatomy of a large-scale hypertextual Web search engine
... This is largely because they all have high PageRanks. However, once the system was running smoothly, S. Brin, L. Page (Computer Networks and ISDN Systems) ... Google employs a number of techniques to improve search quality including page rank, anchor text, and proximity ...

Modeling word burstiness using the Dirichlet distribution
N. Shotton, D. Knowles, D. C. Duin - Proceedings of the 22nd international ... 2005 - dl.ac.m...</n

4 Types of "Rectangle filters" (Similar to Haar wavelet Papageorgiou, et al.)

Based on 24x24 grid:
160,000 features to choose from

\[ g(x) = \text{sum(WhiteArea)} - \text{sum(BlackArea)} \]

"weak" learners

\[ F(x) = \sum \alpha_i f_i(x) \]

\[ f_i(x) = \begin{cases} 1 & \text{if } g_i(x) > \theta_i \\ -1 & \text{otherwise} \end{cases} \]

Example output
Solving other “Face” Tasks

Facial Feature Localization
Profile Detection
Demographic Analysis

“weak” classifiers learned

Bagging vs Boosting

Change in error rate over standard classifier
Ada-Boosting
Arcing
Bagging
White bar represents 1 standard deviation

Boosting Neural Networks

Popular Ensemble Methods: An Empirical Study

Boosting Decision Trees